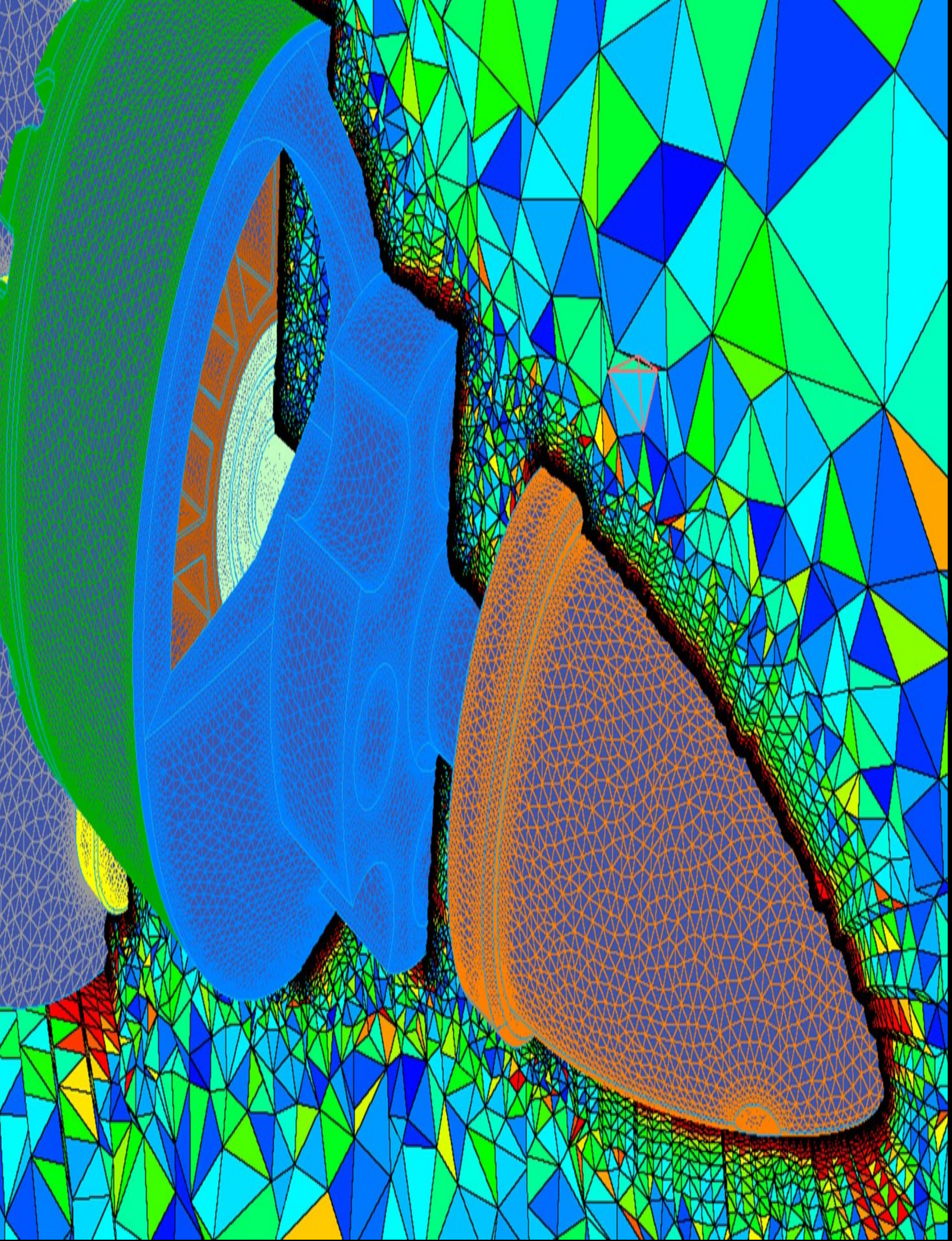


MÉTODO NUMÉRICOS

Aula 5 – Interpolação

Curso de Ciência da Computação

Dr. Rodrigo Xavier de Almeida Leão

Cientista de Dados

Qual a equação da curva que representa os dados?

Qual o valor de L para QS = 1200?

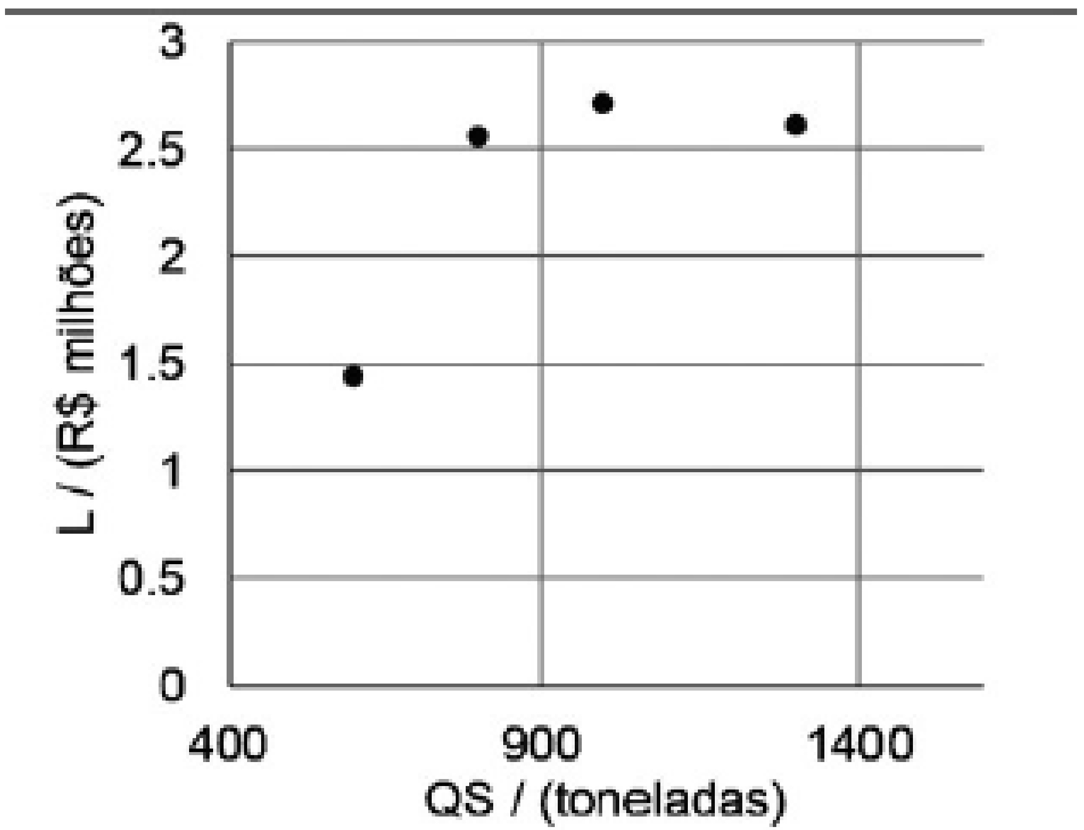
QS (toneladas)	L (milhões de reais)
600	1,43
800	2,55
1000	2,71
1300	2,61

a pela autora

Qual a equação da curva que representa os dados?

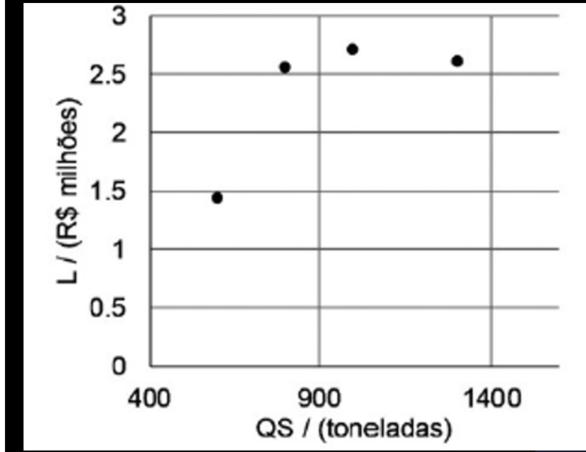
Qual o valor de L para QS = 1200?

Dados de quantidade de silicone em função do lucro



Prof. Dr. Rodrigo Xavier de Almeida Leão

Você terá que realizar a interpolação polinomial sobre os pontos contidos na Figura 3.1 e, aproveitando da equação obtida, responder ao questionamento do gerente com relação ao lucro obtido quando forem produzidas 1200 toneladas de silicone. Como seria possível obter o polinômio interpolador a partir das informações que lhe foram fornecidas? À primeira vista, os pares (QS, L) não têm comportamento linear, tampouco aspectos de uma função de segundo grau. Contudo, um polinômio de grau 3 parece ser adequado ao questionamento que lhe foi feito. Por isso, determine os passos requeridos na interpolação polinomial pelo método de Lagrange e o sistema linear para determinar os coeficientes de $L = a_0 + a_1QS + a_2QS^2 + a_3QS^3$. Além disso, verifique se de fato os polinômios obtidos representam fidedignamente os nós utilizados e quais modificações teriam que ser realizadas para utilizar os algoritmos já vistos para a resolução de sistemas lineares aplicados à interpolação.



Conceito de interpolação

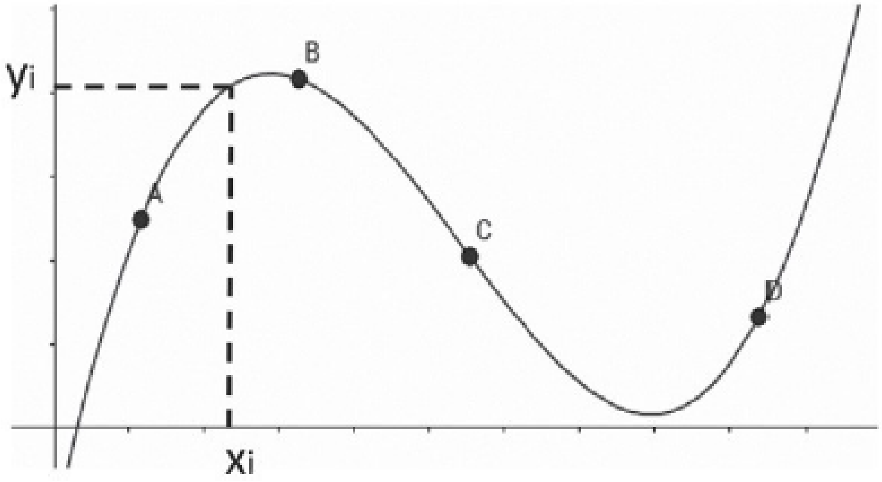
Nos cálculos das mais variadas áreas do conhecimento, nós nos deparamos com as seguintes situações:

- Conhecemos o valor numérico em determinados pontos $(x_0, x_1, x_2, ...)$ com os respectivos $(y_0, y_1, y_2, ...)$, mas desconhecemos a função que os relaciona.
- *f(x)* é difícil de ser manipulada, seja para cálculo de derivada, zero de função ou integral.

Para solucionarmos essas três possibilidades, podemos utilizar métodos de interpolação trigonométrica, exponencial, logarítmica e polinomial. Devido à facilidade de trabalharmos matematicamente com polinômios, nosso foco de estudo será sobre a interpolação polinomial.

A Figura 3.2 ilustra um exemplo em que são conhecidas as coordenadas dos pontos A, B, C e D sobre os quais se aplicou um método para a obtenção do polinômio interpolador, que está representado pela curva cheia, e a partir desse seremos capazes de determinar o valor da variável y_i conhecendo-se x_i.

Figura 3.2 | Ilustração de polinômio interpolador sobre pontos conhecidos



Fonte: adaptada de GeoGebra (2017).

Assim, necessitamos definir terminologias comumente empregadas quando se trabalha com polinômios, as quais estão enunciadas a seguir.

- Polinômio de uma variável é uma série de termos escrita por: $a_n x^n$, sendo: "a" coeficiente; "x" variável independente; "n" expoente natural, isto é:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_1 x + a_0 .$$

- Grau: expoente "n" mais alto entre as variáveis de seus termos não nulos.

- Valor numérico: valor resultante da equação ao se estipular um valor para a variável.

Assim, para o caso em que $p(x) = 2x^4 + 3x - 1$, estamos diante de um polinômio de 4°, grau cujo valor numérico para x=1 é calculado por $p(1) = 2(1)^4 + 3(1) - 1$, que resulta em 4.

Sistema linear gerado pela interpolação

A interpolação polinomial é a determinação de um polinômio a partir de valores conhecidos em certos pontos, denominados nós de interpolação, ou seja, desejamos obter um polinômio de grau no máximo *n*, P_n, para os *n* + 1 pontos distintos $x_0, x_1, x_2, \ldots, x_n$ com os respectivos $y_0, y_1, y_2, \cdots, y_n$. Resumidamente, teremos: $P_n(x_0) = y_0$, $P_n(x_1) = y_1$, ..., $P_n(x_n) = y_n$.

Tabela 3.2 | Pares ordenados (x, f(x))

x	x_0	x_1	x_2	...	x_n
f(x)	$f(x_0)$	$f(x_1)$	$f(x_2)$	...	$f(x_n)$

a_i , $0 \le i \le n$, tal que as igualdades seguintes sejam satisfeitas:

$$
\begin{aligned}
&a_0 + a_1 x_0 + \ldots + a_{n-1} x_0^{n-1} + a_n x_0^n = f(x_0) = y_0 \\
&a_0 + a_1 x_1 + \ldots + a_{n-1} x_1^{n-1} + a_n x_1^n = f(x_1) = y_1 \\
&\vdots \\
&a_0 + a_1 x_n + \ldots + a_{n-1} x_n^{n-1} + a_n x_n^n = f(x_n) = y_n
\end{aligned}
$$

Se observamos, estamos diante da resolução de um sistema linear de ordem n, cujas incógnitas são a_i . Escrevendo no formato $Ax = b$, como definido na Unidade 2:

$$
\begin{cases}
a_0 + a_1 x_0 + \ldots + a_{n-1} x_0^{n-1} + a_n x_0^n = f(x_0) = y_0 \\
a_0 + a_1 x_1 + \ldots + a_{n-1} x_1^{n-1} + a_n x_1^n = f(x_1) = y_1 \\
\vdots \\
a_0 + a_1 x_n + \ldots + a_{n-1} x_n^{n-1} + a_n x_n^n = f(x_n) = y_n
\end{cases}
\Rightarrow Ax = b \Rightarrow
\begin{bmatrix}
1 & x_0 & \cdots & x_0^{n-1} & x_0^n \\
1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
1 & x_{n-1} & \cdots & x_{n-1}^{n-1} & x_{n-1}^n \\
1 & x_n & \cdots & x_n^{n-1} & x_n^n
\end{bmatrix}
\begin{bmatrix}
a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n
\end{bmatrix}
=
\begin{bmatrix}
y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n
\end{bmatrix}
$$

$$Ax = b \Rightarrow \begin{bmatrix} 1 & x_0 & \cdots & x_0^{n-1} & x_0^n \\ 1 & x_1 & \cdots & x_1^{n-1} & x_1^n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{n-1} & \cdots & x_{n-1}^{n-1} & x_{n-1}^n \\ 1 & x_n & \cdots & x_n^{n-1} & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$$

$$A = \begin{bmatrix} x_0^n & x_0^{n-1} & \ldots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \cdots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \cdots & x_n & 1 \end{bmatrix}, \quad x = (a_n \; a_{n-1} \; \ldots \; a_0)^T$$

$b = (y_0 \; y_1 \; \ldots \; y_n)^T$, cuja solução pode ser obtida pelos métodos de resolução de sistema linear, como a eliminação de Gauss, para o qual, inclusive, já definimos um algoritmo.

A matriz A é denominada por matriz de Vandermonde, cujo determinante, det(A), é expresso por: $\det(A) = \prod_{i<j} (x_i - x_j)$ $= (x_1 - x_2)(x_1 - x_3)\ldots(x_{n-1} - x_n)$, como x_i são distintos, $\det(A) \neq 0$, ou seja, o sistema linear possui uma única solução e, portanto, os coeficientes $a_0, a_1, \ldots, a_n$ do polinômio são únicos, o que significa que se obtém apenas um polinômio de grau *n* para os *n* + 1 pontos conhecidos.

EXEMPLO

Em um experimento, foram coletados os seguintes pares ordenados: (-1,4), (0,1) e (2,-1). Como procederíamos realizar uma interpolação polinomial utilizando esses três pontos? Qual seria o grau do polinômio a ser obtido? Verifique se o polinômio interpolador obtido reproduz os pontos conhecidos.

Para respondermos a esses questionamentos, verificamos que são conhecidos três pontos distintos: $x_0 = -1$, $x_1 = 0$ e $x_2 = 2$, com as respectivas ordenadas: $y_0 = 4$, $y_1 = 1$ e $y_2 = -1$.

Diante disso, podemos escrever um polinômio de grau, no máximo, 2, ou seja, $p(x) = a_2 x^2 + a_1 x + a_0$.

Exemplo

Aplicando o método de sistema linear para determinarmos a_2, a_1 e a_0,

temos: $\begin{cases} a_2x_0^2 + a_1x_0 + a_0 = y_0 \\ a_2x_1^2 + a_1x_1 + a_0 = y_1 \\ a_2x_2^2 + a_1x_2 + a_0 = y_2 \end{cases}$.

A matriz de Vandermonde para três pontos é: $A = \begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \end{bmatrix}$, que

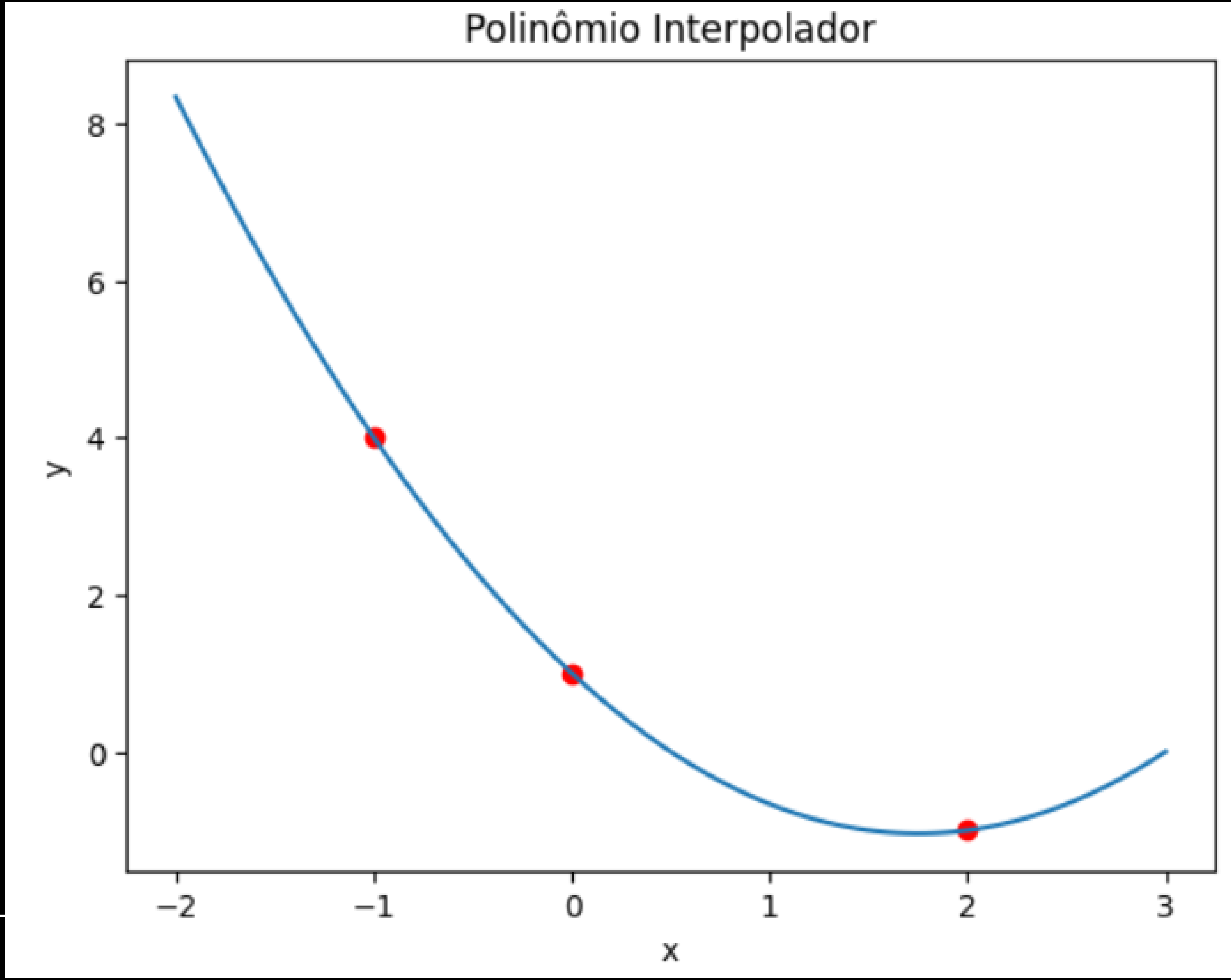
A matriz de Vandermonde para três pontos é: $A = \begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \end{bmatrix}$, que para o exemplo recai em: $A = \begin{bmatrix} (-1)^2 & -1 & 1 \\ 0 & 0 & 1 \\ 2^2 & 2 & 1 \end{bmatrix}$, $x = (a_2 \ a_1 \ a_0)^T$ e $b = (4 \ 1 \ -1)^T$.

$$Ax = b \Rightarrow \begin{bmatrix} (-1)^2 & -1 & 1 \\ 0 & 0 & 1 \\ 2^2 & 2 & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ -1 \end{bmatrix}.$$ O sistema linear

```python
import numpy as np
import matplotlib.pyplot as plt

# Supondo que você já tenha a solução x do sistema linear
# x = [1, 2, 3]  # Substitua pelos valores reais da solução

# Criando um vetor de pontos para plotar o polinômio
x_values = np.linspace(-2, 3, 100)

# Calculando os valores do polinômio
y_values = 0
for i in range(len(x)):
  y_values += x[i] * x_values**(len(x) - 1 - i)

```

```python
# Plotando o polinômio
plt.plot(x_values, y_values)

# Plotando os pontos X          Y   originais
plt.scatter(    [-1, 0, 2], [4, 1, -1], color='red')

# Adicionando rótulos aos eixos e título do gráfico
plt.xlabel('x')
plt.ylabel('y')
plt.title('Polinômio Interpolador')

# Exibindo o gráfico
plt.show()
```

SOLUÇÃO COM x CORRETO



SOLUÇÃO Exemplo 1

Dessa forma, o polinômio obtido é: $p(x) = 0,667x^2 - 2,333x + 1$

$$p(-1) = 0,667(-1)^2 - 2,333(-1) + 1 = 4,$$

$$p(0) = 0,667(0)^2 - 2,333(0) + 1 = 1 \text{ e}$$

$$p(2) = 0,667(2)^2 - 2,333(2) + 1 = -0,992.$$

Encontre o polinômio interpolador

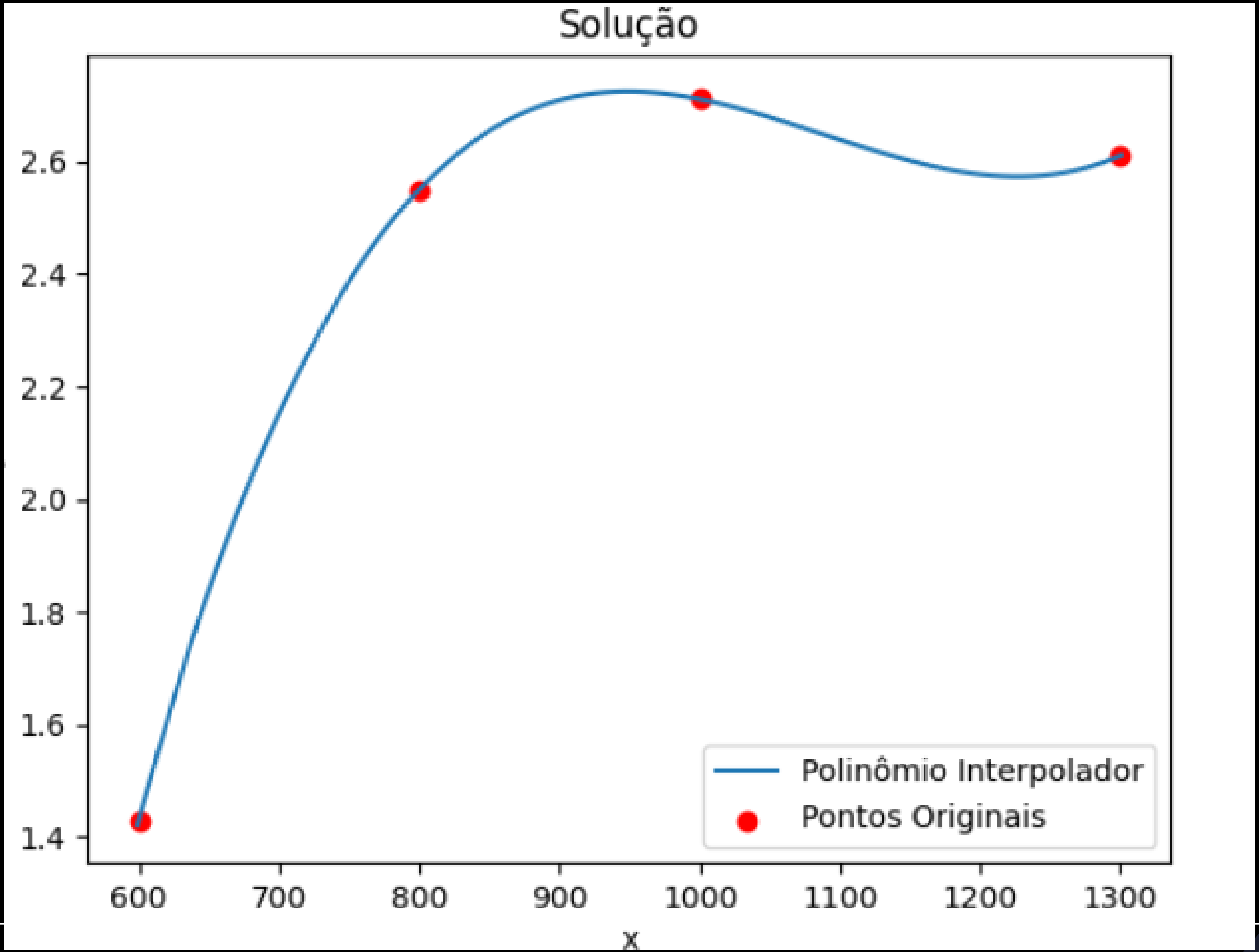
Dados de quantidade de silicone e respectivo lucro

i	x_i	y_i
0	600	1,43
1	800	2,55
2	1000	2,71
3	1300	2,61

Desenhar um polinômio e pontos originais

```python
coefficients = solve_linear_system(A, b)

if coefficients is not None:
  print("Solução do sistema:")
  print(coefficients)
#-----------------------------------------------------------
x_points= [600,800, 1000, 1300]
y_points=[1.43, 2.55, 2.71, 2.61]


# Cria um vetor de pontos para plotar o polinômio
x_values = np.linspace(min(x_points) - 1, max(x_points) + 1, 100)
y_values = np.polyval(coefficients, x_values)

# Plota o polinômio
plt.plot(x_values, y_values, label="Polinômio Interpolador")

# Plota os pontos originais
plt.scatter(x_points, y_points, color='red', label="Pontos Originais")
```

```python
# Adiciona legenda, título e rótulos aos eixos
plt.legend()
plt.title("Solução")
plt.xlabel("x")
plt.ylabel("y")

# Exibe o gráfico
plt.show()

# Tabela de iteração (opcional)
print("\nTabela de Iteração:")
print("--------------------")
print("i | x_i | y_i")
print("--------------------")
for i in range(len(x_points)):
    print(f"{i} | {x_points[i]} | {y_points[i]}")
```


Solução
2.6
2.4
2.2
2.0
1.8
1.6
1.4
600
700
800
900
1000
1100
1200
1300
x
Polinômio Interpolador
Pontos Originais

Fórmula de Lagrange

Outro modo de determinarmos o polinômio interpolador de grau n é a partir da fórmula de Lagrange, que requer $x_0, x_1, x_2, \ldots, x_n$, isto é, $n + 1$ pontos distintos e a determinação de funções L_k, com k = 0, 1, ..., n, o que resulta em $p_n(x) = y_0 L_0(x) + y_1 L_1(x) + \ldots + y_n L_n(x)$. Os L_k são funções polinomiais, tais que:

$$L_k(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{k-1})(x - x_{k+1})\ldots(x - x_n)}{(x_k - x_0)(x_k - x_1)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n)},$$

com $L_k(x_i) = \begin{cases} 0 \text{ se } k \neq i \\ 1 \text{ se } k = i \end{cases}$.

```
import numpy as np
import matplotlib.pyplot as plt

def lagrange_interpolation(x_points, y_points):
    """
    Calcula o polinômio interpolador usando a interpolação de Lagrange.
    Args:
        x_points: Lista de coordenadas x dos pontos.
        y_points: Lista de coordenadas y dos pontos.
    Returns:
        Uma lista de coeficientes do polinômio interpolador.
    """

    n = len(x_points)
    coefficients = np.zeros(n)
    for i in range(n):
        term_coefficients = [1]  # Começa com coeficiente 1
        for j in range(n):
            if i != j:
                # Multiplica por (x - xj) / (xi - xj)
                term_coefficients = np.polymul(term_coefficients,
                                               np.array([1, -x_points[j]]))
                term_coefficients = np.polymul(
                    term_coefficients, 1.0 / (x_points[i] - x_points[j]))

        coefficients += y_points[i] * term_coefficients

    return coefficients
```